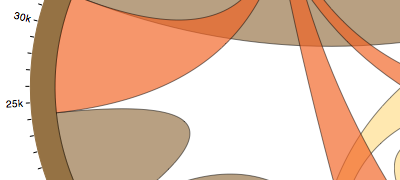
Chord 输出

**Chord diagrams** show relationships among a group of entities. For example, consider a hypothetical population of people with different hair colors: black, blonde, brown and red. Each person in this population has a preferred hair color for a dating partner; of the 29,630 (hypothetical) people with black hair, 40% (11,975) prefer partners with the same hair color. This preference is asymmetric: for example, only 10% of people with blonde hair prefer black hair, while 20% of people with black hair prefer blonde hair.

Chord图所示内容为一组实体之间的关系。例如，假定有一组不同发色的人：黑色、金色、棕色和红色。 该组每个人都为约会对象准备了中意的发色；在所有29630个（假设）黑发人群中， 40%的人（11975）倾向于同具有相同发色的对象进行约会。但这种倾向是非对称的：例如，只有10%的金发人会选择黑发人群作为约会对象，而20%的黑发人会将金发人作为约会对象。

[](http://mbostock.github.com/d3/ex/chord.html)

A chord diagram visualizes these relationships by drawing quadratic Bézier curves between arcs. The source and target arcs represents two mirrored subsets of the total population, such as the number of people with black hair that prefer blonde hair, and the number of people with blonde hair that prefer black hair. As another example, consider this chord diagram of [software dependencies](http://bl.ocks.org/mbostock/1046712).

The chord layout is designed to work in conjunction with the [chord shape](https://github.com/mbostock/d3/wiki/SVG-Shapes#wiki-chord) and the [arc shape](https://github.com/mbostock/d3/wiki/SVG-Shapes#wiki-arc). The layout is used to generate data objects which describe the chords, serving as input to the chord shape. The layout also generates descriptions for the groups, which can be used as input to the arc shape.

通过在不同的弧线之间画出二次贝塞尔曲线，将上述关系表示在一张chord图中。源弧线和目标弧线分别代表总人口的两个镜像子集，如喜欢金发的黑发人口数量，以及喜欢黑发的金发人口数量。另一个例子，我们可以 chord图[software dependencies](http://bl.ocks.org/mbostock/1046712) （软件依赖）。

chord布局同chord shape(chord 形状)和arc shape(弧形)协同工作，用于生成数据对象。该对象作为chord形状的输入，对chord进行描述。同时，该布局还能生成对不同群组的描述，用作弧形的输入。

#### [#](https://github.com/mbostock/d3/wiki/Chord-Layout#wiki-chord) d3.layout.chord()

Constructs a new chord layout. By default, the input data is not sorted, and there is no padding between groups. Unlike some of the other layouts, the chord layout is not a function to be applied to data; instead, data is specified by setting the associated [matrix](https://github.com/mbostock/d3/wiki/Chord-Layout#wiki-matrix), and retrieved using the [chords](https://github.com/mbostock/d3/wiki/Chord-Layout#wiki-chords) and [groups](https://github.com/mbostock/d3/wiki/Chord-Layout#wiki-groups) accessors.

#### [#](https://github.com/mbostock/d3/wiki/Chord-Layout#wiki-chord) d3.输出.chord()

构建新的chord布局。在默认情况下，输入数据并未分类，并且各群组之间没有填充。和其他布局不同，该chord布局并不是应用于数据的函数；相反，数据通过设置关联矩阵来指定，通过chords和groups存取器检索。

#### [#](https://github.com/mbostock/d3/wiki/Chord-Layout#wiki-matrix) chord.matrix([matrix])

If matrix is specified, sets the input data matrix used by this layout. If matrix is not specified, returns the current data matrix, which defaults to undefined. The input matrix must be a [square matrix]([http://en.wikipedia.org/wiki/Matrix\_(mathematics\)#Square\_matrices](http://en.wikipedia.org/wiki/Matrix_%28mathematics%5C%29#Square_matrices)) of numbers, such as:

[[11975, 5871, 8916, 2868],

[ 1951, 10048, 2060, 6171],

[ 8010, 16145, 8090, 8045],

[ 1013, 990, 940, 6907]]

Each row in the matrix corresponds to a distinct group, such as a hair color in the above example. Each column i in the matrix corresponds to the same group as row i; the cell ij corresponds to the relationship from group i to group j.

#### [#](https://github.com/mbostock/d3/wiki/Chord-Layout" \l "wiki-matrix) chord.矩阵([矩阵])

指定*矩阵*之后，设定该布局用到的输入数据矩阵。如果没有指定矩阵，返回当前数据矩阵，默认为未定义。输入矩阵的数字必须为“方形矩阵”

([http://en.wikipedia.org/wiki/Matrix\_(mathematics\)#Square\_matrices](http://en.wikipedia.org/wiki/Matrix_%28mathematics%5C%29#Square_matrices)) 例如：

[[11975,5871,8916,2868],

[1951,10048,2060,6171],

[8010,16145,8090,8045],

[1013,990,940,6907]]

矩阵的每一行对应一个特定分组，如上文所述某个发色。 矩阵中每一列*i* 同第*i* 行相对应；每个单元格ij 对应表示第i组到第j组之间的关系。

[#](https://github.com/mbostock/d3/wiki/Chord-Layout" \l "wiki-padding) chord.填充([填充])

指定*填充*之后，在不同组之间设定角度填充，并为其在radians中指定值。 如果没有指定*填充*，返回当前填充，默认值为0. 你可能希望根据分组数量计算填充（关联矩阵中行和列的数量）。

#### [#](https://github.com/mbostock/d3/wiki/Chord-Layout#wiki-sortGroups) chord.sortGroups([comparator])

If comparator is specified, sets the sort order of groups (rows) for the layout using the specified comparator function. The comparator function is invoked for pairs of rows, being passed the sum of row i and row j. Typically, the comparator should be specified as either [d3.ascending](https://github.com/mbostock/d3/wiki/Arrays#wiki-d3_ascending) or [d3.descending](https://github.com/mbostock/d3/wiki/Arrays#wiki-d3_descending). If comparator is not specified, returns the current group sort order, which defaults to null for no sorting.

#### [#](https://github.com/mbostock/d3/wiki/Chord-Layout" \l "wiki-sortGroups) chord.sortGroups ([comparator])

如果已经指定*comparator*，使用指定comparator函数为布局设定分组（行）的排列顺序。当行成对出现时，调用comparator函数， 值为行*i*和行*j*的总和。通常，需要将comparator按照[d3.ascending](https://github.com/mbostock/d3/wiki/Arrays#wiki-d3_ascending)或[d3.descending](https://github.com/mbostock/d3/wiki/Arrays#wiki-d3_descending)进行指定。如果没有指定comparator，则返回当前分组排列顺序，该顺序默认值为空。

#### [#](https://github.com/mbostock/d3/wiki/Chord-Layout#wiki-sortSubgroups) chord.sortSubgroups([comparator])

If comparator is specified, sets the sort order of subgroups (columns within rows) for the layout using the specified comparator function. The comparator function is invoked for pairs of cells, being passed the value of each cell. Typically, the comparator should be specified as either ascending or descending. If comparator is not specified, returns the current subgroup sort order, which defaults to null for no sorting.

#### [#](https://github.com/mbostock/d3/wiki/Chord-Layout" \l "wiki-sortSubgroups) chord.sortSubgroups([comparator])

如果已经指定*comparator,* 使用指定comparator函数为布局设定分组（行内各列）的排列顺序。当单元格成对出现时，调用comparator函数， 值为各单元格的值。通常，需要将comparator以升序或降序进行指定。如果没有指定comparator，则回当前子分组排列顺序，该顺序默认值为空。

#### [#](https://github.com/mbostock/d3/wiki/Chord-Layout#wiki-sortChords) chord.sortChords([comparator])

If comparator is specified, sets the sort order of chords (z-order) for the layout using the specified comparator function. The comparator function is invoked for pairs of chords, being passed the minimum value of the associated source and target cells. Typically, the comparator should be specified as either ascending or descending. If comparator is not specified, returns the current chord sort order, which defaults to null for no sorting.

null for no sorting.

#### [#](https://github.com/mbostock/d3/wiki/Chord-Layout" \l "wiki-sortChords) chord.sortChords([comparator])

如果已经指定*comparator,* 运用指定comparator函数为布局设定chord （Z顺序）的排列顺序。当chord成对出现时，调用comparator函数，值为源单元格和目标单元格的最小值。通常，要将comparator以升序或降序进行指定。如果没有指定comparator，返回当前chord排列顺序，默认值为空。

#### [#](https://github.com/mbostock/d3/wiki/Chord-Layout#wiki-chords) chord.chords()

Returns the computed chord objects, given the layout's current configuration and associated matrix. If the chord objects were previously-computed, this method returns the cached value. Changing any attribute of the layout implicitly clears the previously-computed chords, if any, such that the next call to this method will recompute the layout. The returned objects have the following properties:

* source - an object describing the source.
* target - an object describing the target.

These objects, in turn, describe the underlying entity:

* index - the row index, i.
* subindex - the column index, j.
* startAngle - the start angle of the arc, in radians.
* endAngle - the end angle of the arc, in radians.
* value - the value of the associated cell ij, a number.

Note that these objects conveniently match the default accessors for the [chord](https://github.com/mbostock/d3/wiki/SVG-Shapes#wiki-chord) generator; however, you can still override the accessors to tweak the layout, or simply manipulate the returned objects.

#### [#](https://github.com/mbostock/d3/wiki/Chord-Layout" \l "wiki-chords) chord.chords()

返回计算过的chord对象，设当前布局的配置和关联矩阵。如果chord对象已计算完毕，本方法返回缓存值。如果布局属性有任何改变，则清空之前计算的chord。此时，如果下次调用该方法，需要对布局进行重新计算。返回对象具有下列属性：

* 源-描述源对象。
* 目标-描述目标对象。

这两个对象描述下列实体：

* 索引-行索引，*i*。
* 子索引-列索引，*j*。
* 起始角-弧的起始角， 在radians内。
* 终止角-弧的终止角，在radians内。
* 值-关联单元格*ij*的数值。

需要注意的是，这些对象同chord生成器的默认存取器具有较好的吻合度；但仍可以对存取器进行重写或者修改返回对象，实现布局微调。

#### [#](https://github.com/mbostock/d3/wiki/Chord-Layout#wiki-groups) chord.groups()

Returns the computed group objects, given the layout's current configuration and associated matrix. If the group objects were previously-computed, this method returns the cached value. Changing any attribute of the layout implicitly clears the previously-computed groups, if any, such that the next call to this method will recompute the layout. The returned objects have the following properties:

* index - the row index, i.
* startAngle - the start angle of the arc, in radians.
* endAngle - the end angle of the arc, in radians.
* value - the sum of the associated row i, a number.

Note that these objects conveniently match the default accessors for the [arc](https://github.com/mbostock/d3/wiki/SVG-Shapes#wiki-arc) generator; however, you can still override the accessors to tweak the layout, or simply manipulate the returned objects.

#### [#](https://github.com/mbostock/d3/wiki/Chord-Layout" \l "wiki-groups) chord.groups()

返回计算过的分组对象，设当前布局的配置和关联矩阵。如果分组对象已计算完毕，本方法返回缓存值。如果布局属性有任何改变，则清空之前计算的分组。此时，如果下次调用该方法，需要对布局进行重新计算。返回对象具有下列属性：

* 索引-行索引，*i*。
* 起始角-弧的起始角， 在radians内。
* 终止角-弧的终止角，在radians内。
* 值-相关行 *i*的值的总和。

需要注意的是，这些对象同arc生成器的默认存取器具有较好的吻合度；但仍可以对存取器进行重写或者修改返回对象，实现布局微调。

### [Cluster](https://github.com/mbostock/d3/wiki/Cluster-Layout)

* [d3.layout.cluster](https://github.com/mbostock/d3/wiki/Cluster-Layout#wiki-cluster) - cluster entities into a dendrogram.
* [cluster](https://github.com/mbostock/d3/wiki/Cluster-Layout#wiki-_cluster) - alias for cluster.nodes.
* [cluster.nodes](https://github.com/mbostock/d3/wiki/Cluster-Layout#wiki-nodes) - compute the cluster layout and return the array of nodes.
* [cluster.links](https://github.com/mbostock/d3/wiki/Cluster-Layout#wiki-links) - compute the parent-child links between tree nodes.
* [cluster.children](https://github.com/mbostock/d3/wiki/Cluster-Layout#wiki-children) - get or set the accessor function for child nodes.
* [cluster.sort](https://github.com/mbostock/d3/wiki/Cluster-Layout#wiki-sort) - get or set the comparator function for sibling nodes.
* [cluster.separation](https://github.com/mbostock/d3/wiki/Cluster-Layout#wiki-separation) - get or set the spacing function between neighboring nodes.
* [cluster.size](https://github.com/mbostock/d3/wiki/Cluster-Layout#wiki-size) - get or set the layout size in x and y.
* [cluster.nodeSize](https://github.com/mbostock/d3/wiki/Cluster-Layout#wiki-nodeSize) - specify a fixed size for each node.

### [Cluster](https://github.com/mbostock/d3/wiki/Cluster-Layout) 集群

* [d3.layout.cluster](https://github.com/mbostock/d3/wiki/Cluster-Layout#wiki-cluster) -集群实体树状图。
* [cluster](https://github.com/mbostock/d3/wiki/Cluster-Layout#wiki-_cluster) -集群.节点的别名。
* [cluster.nodes](https://github.com/mbostock/d3/wiki/Cluster-Layout#wiki-nodes) - 计算集群布局，并返回节点阵列。
* [cluster.links](https://github.com/mbostock/d3/wiki/Cluster-Layout#wiki-links) -计算树节点之间的父子链接。
* [cluster.children](https://github.com/mbostock/d3/wiki/Cluster-Layout#wiki-children) -为子节点获取或设定存取器函数。
* [cluster.sort](https://github.com/mbostock/d3/wiki/Cluster-Layout#wiki-sort) - 为同级节点获取或设定存取器函数。
* [cluster.separation](https://github.com/mbostock/d3/wiki/Cluster-Layout#wiki-separation) -为相邻节点获取或设定间隔行数。
* [cluster.size](https://github.com/mbostock/d3/wiki/Cluster-Layout#wiki-size) -在x轴和y轴获取或设置布局大小。
* [cluster.nodeSize](https://github.com/mbostock/d3/wiki/Cluster-Layout#wiki-nodeSize) -为每个节点指定固定值。